Comparison of a Local Linearization Algorithm with Standard Numerical Integration Methods for Real-Time Simulation

Cook, Gerald; Lin, Ching-Fang

Published in:
I E E E Transactions on Industrial Electronics

Link to article, DOI:
10.1109/TIECI.1980.351664

Publication date:
1980

Document Version
Publisher's PDF, also known as Version of record

Comparison of a Local Linearization Algorithm with Standard Numerical Integration Methods for Real-Time Simulation

GERALD COOK, SENIOR MEMBER, IEEE, AND CHING-FANG LIN

Abstract—**The local linearization algorithm is presented as a possible numerical integration scheme to be used in real-time simulation. A second-order nonlinear example problem is solved using different methods. The local linearization approach is shown to require less computing time and give significant improvement in accuracy over the classical second-order integration methods.**

I. Introduction

THERE have been extensive studies made to obtain more accurate less time-consuming techniques for approximately transforming continuous systems to discrete systems. This area is especially important for real-time simulation applications.

Selection of a particular method would depend on the specific simulation problem. It is a general rule, however, that there is a trade off between speed and accuracy. In other words, a more accurate discretization method is in general more time-consuming. For example, integration by a Runge-Kutta (RK) method, which is obtained by truncating a Taylor series expansion of the function, can provide an approximation which closely matches the ideal integration when a small step size T is used. The fourth-order Runge-Kutta method (RK-4) truncates the expansion after the term proportional to T^4. Therefore the truncation error in this case is proportional to T^5. This method requires four evaluations of the system equations at each sample point, however, and so is generally too time-consuming for many real-time applications.

In the search for a fast and accurate discrete-time simulation method a general algorithm, based on a local linearization procedure, has been developed. This algorithm is extremely simple and thus relatively fast. In the proceeding sections this algorithm will be developed and then applied to an example problem. Comparisons made with other numerical methods regarding speed and accuracy indicate that this is a very promising algorithm.

II. Local Linearization Algorithm

Let the nth-order ordinary differential equation, describing a general nonlinear time-varying system, be expressed in state variable form as

$$\dot{x}(t) = f(x, u) \tag{1}$$

where $x(t)$ is the n-dimensional vector representing the system states and f is the n-dimensional vector composed of general nonlinear time-invariant functions of the state vector $x(t)$ and the r-dimensional control vector $u(t)$.

When (1) is expressed as a Taylor series in $x(t)$ about the point $x(t_k)$ or x_k one obtains

$$\dot{x} = f\left(x_k, u(t) + \frac{\partial f}{\partial x}(x_k, u(t))\right) \cdot (x(t) - x_k) + \cdots \tag{2}$$

or

$$\dot{x} = Ax + g(x_k, u(t)) + \cdots \tag{3}$$

where

$$A = \frac{\partial f}{\partial x}(x_k, u)$$

and

$$g(x_k, u) = f(x_k, u) - \frac{\partial f}{\partial x}(x_k, u) \cdot x_k.$$

When (3) is integrated, it becomes

$$x_{k+1} = [\exp(AT)]x_k + \left[\int_{t_k}^{t_{k+1}} \exp[A(t_{k+1} - \tau)] \cdot g(x_k, u(\tau))\, d\tau\right] \tag{4}$$

where

$$T = t_{k+1} - t_k.$$

Let us assume that our integration interval is such that

$$u(t) \approx u(t_k), \qquad t_k \leqslant t < t_{k+1}. \tag{5}$$

Almost all numerical integration methods require this to be the case. RK-4 does sample the input signal twice for each integration interval so the time interval for (5) would be halved for that case.

With the assumption (5), equation (4) becomes

$$x_{k+1} = \exp(AT)x_k + \left[\int_{t_k}^{t_{k+1}} \exp[A(t_{k+1} - \tau)]\, d\tau\right] \cdot g(x_k, u_k) \tag{6}$$

Manuscript received October 24, 1978; revised February 29, 1980.

G. Cook is with the Control Engineering Laboratory, Technical University of Denmark, Copenhagen, Denmark, on leave from the Department of Electrical Engineering, University of Virginia, Charlottesville, VA 22901.

C-F. Lin was with the Department of Electrical Engineering, University of Virginia, Charlottesville, VA. He is now with the Department of Mathematics, University of Michigan, Ann Arbor, MI 48104.

0018-9421/80/0800-0129$00.75

or

$$x_{k+1} = \exp(AT)x_k + \left[\int_0^T \exp(A\tau)\,d\tau\right] \cdot [f(x_k, u_k) - Ax_k] \tag{7}$$

or

$$x_{k+1} = \exp(AT)x_k - A\int_0^T \exp(A\tau)\,d\tau\, x_k + \int_0^T \exp(A\tau)\,d\tau\, f(x_k, u_k). \tag{8}$$

But

$$-A\int_0^T \exp(A\tau)\,d\tau = -\exp(AT) + I.$$

Therefore (8) becomes

$$x_{k+1} = x_k + \left[\int_0^T \exp(A\tau)\,d\tau\right] f(x_k, u_k). \tag{9}$$

It should be noted that (9) implies that $f(x_k, u_k)$ is updated at each sample time. However, A and the integral of $\exp(A\tau)$ may be updated less frequently depending on the severity of the nonlinearity.

The basis for this surprisingly simple algorithm is given by Barker *et al.* [1]. The purpose of this paper is to present the method and to compare it with some standard numerical integration algorithms.

III. Computational Considerations

The most time-consuming calculation for the local linearization algorithm is the updating of the integral of the matrix exponential

$$\int_0^T \exp(A\tau)\,d\tau = IT + AT^2/2! + A^2T^3/3! + \cdots. \tag{10}$$

For convergence one should take enough terms (N) in this series such that

$$\lambda_{\max}\, T/N << 1 \tag{11}$$

or

$$N > 10\lambda_{\max}\, T \tag{12}$$

where $\lambda_{\max}$ is the magnitude of that eigenvalue of A with greatest magnitude. Choosing N in this manner guarantees that the last term retained in the series (10) will be small compared to the previous term.

Defining T_1 as the interval between updates of (10), this calculation requires n^3N multiplications and n^3N additions every T_1 seconds. We shall assume that multiplication requires two units of computing time and addition one unit. Thus calculation of (10) requires $3n^3N/T_1$ units per second of simulation time.

Let us define m as the average number of units of computing time required to update each component of the vector $f(x_k, u_k)$. Then to calculate the A matrix would require $mn^2 + 3n^2$ of computing time. This too would be done every T_1 seconds or would result in an additional $(mn^2 + 3n^2)/T_1$ units per second of simulation time.

Finally to implement (9) would require the nm units to update $f(x_k, u_k)$, n^2 multiplications, and $n^2 + n$ additions. This adds up to $(nm + 3n^2 + n)/T$ units per second of simulation time.

The total time for the local linearization algorithm then becomes

$$N_{LL} = (3n^3N + 3n^2 + mn^2)/T_1 + (nm + 3n^2 + n)/T. \tag{13}$$

For the Euler method

$$x_{k+1} = x_k + Tf(x_k, u_k). \tag{14}$$

The number of computing units per second of simulation time is

$$N_E = (nm + 3n)/T. \tag{15}$$

For the second-order Adams-Bashforth (AB-2) method

$$x_{k+1} = x_k + T(\tfrac{3}{2}f(x_k, u_k) - \tfrac{1}{2}f(x_{k-1}, u_{k-1})) \tag{16}$$

the number of computing units per second of simulation time is

$$N_{\mathrm{AB}} = (nm + 6n)/T. \tag{17}$$

For the RK-4 method

$$x_{k+1} = x_k + \tfrac{1}{6}T(k_1 + 2k_2 + 2k_3 = k_4) \tag{18}$$

the number of computing units per second of simulation time is

$$N_{\mathrm{RK}} = (4nm + 10n)/T. \tag{19}$$

It should be noted that the integration step size required will in general be different for the different methods. Thus T in (13), (15), (17), and (19) is not necessarily the same. Also note that one can immediately dismiss the Euler method by comparing it with the AB-2 method. The number of computations per step size given by (15) and (17) are very nearly equal while generally the step size T required by the Euler method is significantly smaller than that required by the AB-2 method. Thus the Euler method will not be considered further. Further comparisons, e.g., equations (17) and (19) indicate that unless RK-4 can utilize a step-size four times as great as AB-2 it will be more time consuming than AB-2. Experience has indicated that RK-4 cannot use a step size four times greater than AB-2. Therefore, RK-4 can also be eliminated in the competition for real-time simulation methods.

The next section will compare the remaining two integration methods, local linearization and AB-2, via an example problem. RK-4 will be used with a very small integration interval to establish a standard for comparison.

IV. Example Problem

The example to be used for comparing the integration methods is

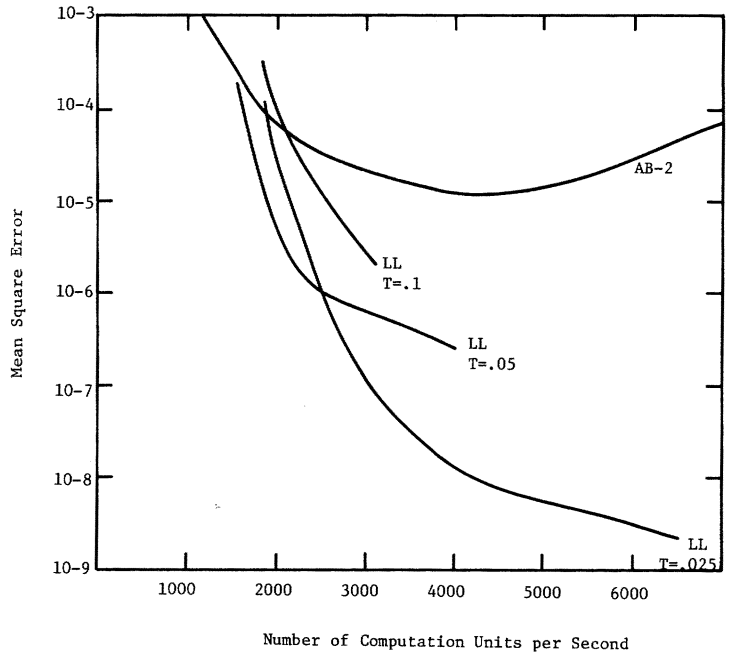

Fig. 1. Comparison of accuracy versus computer requirements for different algorithms and step sizes.

$$\ddot{\theta} + (\dot{\theta})^3 + (1 - \exp(-1\theta 1))\,\mathrm{sgn}\,(\theta) = 0$$

which is modeled in state variable form as

$$\dot{x}_1 = \dot{x}_2$$

$$\dot{x}_2 = -x_2{}^3 - (1 - \exp(-|x_1|)\,\mathrm{sgn}\,(x_1).$$

Initial conditions are

$$x_1(0) = 10$$

and

$$x_2(0) = 0.$$

It was found that the major portion of transient lasted only 1s. Therefore, comparisons will be restricted to this time interval.

For the "ideal" solution, RK-4 was used with a step size of 0.01 s. Taking m to be 10, this means that 10 000 computing units per second were required. We will assume that the mean-squared error for this case is zero.

To use the local linearization algorithm it was decided to try several different combinations of values for T and T_1. This was done by selecting T and varying T_1 over a range of values. First T was set at 0.025. The largest eigenvalue for A was in the neighborhood of 10 which means

$$10\lambda T = 2.5$$

so

$$N = 3.$$

This yields a value for N_{LL} of $(124/T_1 + 34/T)$ or $(124/T_1 + 1360)$ computing units per second.

Another value selected for T was 0.05 s. For this value,

$$10\lambda T = 5$$

so

$$N = 5.$$

This yields a value of N_{LL} of $(172/T_1 + 34/T)$ or $(172/T_1 + 680)$ computing units per second.

Finally T was set equal to 0.1 s. For this value

$$10\lambda T = 10$$

so

$$N = 10$$

and N_{LL} became $(292/T_1 + 34/T)$ or $(292/T_1 + 340)$ computing units per second. Fig. 1 shows graphs of mean-square error, defined as

$$\mathrm{MSE} = \frac{1}{t_f}\int_0^{t_f} [x_1(t) - x_{1A}(t))^2 + (x_2(t) - x_{2A}(t))^2]\,dt$$

versus computation units per second. The $x_{1A}(t)$ and $x_{2A}(t)$ represent the ideal solution obtained by RK-4 and as stated earlier t_f was taken as 1 s. For the curve labeled $T = 0.025$, T_1 was varied and as would be expected, the smaller T_1 was (and the more computation units per second) the smaller was the mean-square error. The end of the curve corresponds to a value for T_1 of 0.025 s., i.e., equal to T. This is the smallest period one could use between updates of A and the integral of $\exp(A\tau)$.

The curves for T taking on values of 0.05 and 0.1 s are also shown. It is interesting to note that for a given number of computation units per second, certain combinations of values for T and T_1 are better than others. For example suppose one can tolerate 3000 computations units per second. Table I shows the various combinations of T and T_1 studied which require 3000 computation units per second and the resulting mean-squared error. This is the kind of information one would like in order to optimize a simulation for a given

TABLE I
COMPARISON OF T, T_1 AND MSE FOR 3000 COMPUTATION UNITS PER SECOND

T	T_1	MSE
.025	.076	1×10^{-7}
.050	.074	8×10^{-7}
.100	.110	3×10^{-6}

TABLE II
COMPARISON OF T, T_1 AND COMPUTATION UNITS PER SECOND FOR MSE $= 1 \times 10^{-5}$

T	T_1	CU/s
.025	.190	2013
.050	.162	1740
.100	.136	2488

computing capability. Obviously the first combination is the most accurate.

Conversely one could have a mean-squared error specification and wish to achieve this performance with a minimum computing capability. Table II shows the kind of comparison one would make if for example the specification on mean squared errors was 10^{-5}. Clearly the combination of

$T = 0.050$ s

and

$T_1 = 0.162$ s

is the most efficient.

In addition to illustrating the behavior of the local linearization method, Fig. 1 also shows the performance versus computation requirements for the AB-2 method. The AB-2 method required a value for N_{AB} or $32/T$ computing units per second. Although this method is very simple to use and is definitely a fine method, the local linearization method outperforms it for this particular example. It is interesting to note that the AB-2 method experiences an extremum in mean-squared error. This is due to growth of the round off error caused by the large number of steps required as T becomes small. This minimum occurs at 4444 computation units per second which corresponds to

$T = 0.0072$ s.

For the local linearization method, values for T this small were not used. No doubt there does exist a point of diminishing returns for the local linearization method just as for any other numerical integration method.

V. CONCLUSION

A local linearization method for discrete-time simulation of nonlinear time-invariant systems has been presented and tested. Computational considerations have been treated. Comparisons with other well-known methods indicate that this is a very promising method, and depending on the problem under consideration may be the best approach to use.

REFERENCES

[1] L. E. Barker, R. L. Bowles, and L. H. Williams, "Development and application of a local linearization algorithm for the integration of quaternian rate equations in real-time flight simulation problems," NASA Tech. Note TN D-7347, Dec. 1973.